Art analysis to import publications

This has been written in 2025.

The main objective is to retrieve some information about a publication and then to store it in the *R3MOB SQL* database.

Where to search - Academic Databases

There are a lot of databases around the world that store scientific publications.

Among themselves, there are for instance:

- ***www.science.gov***: a *U.S government* database that contains scientific research from *U.S federal agencies.*

This one contains more than *200 million pages* of scientific information for free, and is a gateway to over 2000 scientific websites. However, it is centered on the *U.S.*

- ***https://csxstatic.ist.psu.edu/home*** (*CiteSeer*): centered on computer and information science.
- ***https://ieeexplore.ieee.org/Xplore/home.jsp*** (*IEEE Xplore*): very famous, however, it requires a subscription.
- ***https://www.springer.com*** (*Springer*): historical database, and also requires a subscription.

Anyway, you can follow **this link** to get a full list of academic databases.

Among those databases, there are the one which are:

- **Not Free**: import a publication from it will be hard.
- **Free**: import a publication from it will be possible.

So the idea is to search for the publication in the right database. The main idea is to find something that can make request to all those databases.

How to search - Search Engine

The databases discussed above also contain a *search engine.* From the user perspective, it is a *research bar.*

Indexer and Search Library

The *Search library* is the *core* of the search engine. It contains all the algorithmic stuff to be as efficient and accurate as possible.

For instance, let's talk about ***Apache Lucene***. This is an *open source Java* library providing indexing and search features, such as spellchecking, highlighting and advanced tokenization capabilities.

It is used by websites like *Twitter*, *Apple*, and even *Wikipedia*.

However, *Lucene* does not contain *crawling* and *HTML parsing* functionalities. That's why it is not recommended to use it directly.

Lucene-based Search Engines

Many services already extended *Lucene*'s capabilities of parsing, such as:

- ***Apache Nutch***: a *web crawler*, It provides *web crawling* and *HTML* parsing. **Wikipedia** says:

  > Web crawler, sometimes called a spider or spiderbot and often shortened to crawler, is an Internet bot that systematically browses the World Wide Web and that is typically operated by search engines for the purpose of Web indexing.

For instance, the *GoogleBot* is *Google*'s generic web crawler that is responsible for crawling sites that will show up on *Google*'s search engine, ***Kinsta.com*** says.

Another example could be **citeseerxbot**, the crawler of ***CiteSeerX***, discussed above.

- ***Apache Solr***: an 2004 *enterprise-search* platform. It can also *crawl* some web content as well as public databases. This is exactly what it is needed to search for publications.

It is used by ***Apache Hadoop*** which is a very famous framework for *distributed computing*.

- ***ElasticSearch***: also an *enterprise-search* platform, but this one has been released in 2010. It is currently the most popular *search engine*, even if it seems that it is only working using *json* format.

Differences between Solr and ElasticSearch

Both **ElasticSearch** and **Apache Solr** are very powerful. However, the *ElasticSearch API* is only working using *json* format.

ElasticSearch is actually replacing every *Solr*-based app thanks to its ability to integrate *NLP* (Natural Language Processing) and, for instance, **BERT**.

Here is ***a more detailed explanation***.

Which tools are designed to search specifically publications

The *search engines* discussed above are already great to use. They give access to a *REST Api*, with nice documentation. However, they are too generic, they are not configured for especially *scientific publications*.

That's why I'll introduce here mostly two services that are based on a *search engine* and that can be used to search for publications among public databases.

1. ***Crossref***: A *RESTFUL Api* that has been recently updated to use *ElasticSearch* instead of *Solr*. That's why some of the content is a little bit deprecated, but it seems to be the best option for this project.

It is specialized on *research objects*. You can **try it here**.

It is so impressive because *ElasticSearch* is based on a *Peer 2 Peer* architecture. *Crossref* gathers more than *22,000* members across the world, with nearly *2 billion monthly API queries*.

One difficulty could be to *construct the DOIs* of a broken publication if the user wants to import a publication with, actually, a broken *DOI*. Indeed, the *DOIs* (example: *10.1037/0003-066x.59.1.29*) could break a link, because of their format. So this type of error will be hard to solve, a publication may not be retrieved if the user gives a *DOI*.

2. ***Semantic Scholar***: A *REST Api*. Its corpus gathers *214 Million* papers and more than *79 Million* authors.

It is possible to get an *Api key* **right here**. Without an *Api key*, the limitation is set to *1,000* requests per seconds. However, *Semantic Scholar* does not *crawl/index* for material that is behind a *paywall*, **wikipedia** says.

This one is very powerful because it is natively based on *NLP*. The main advantadge compared to *Crossref* is its capability to analyse the content.

You can try it **right here**.

My plan to import publications on the website

There are some publications that are not available freely on the internet. One may take the example of *IEEE Xplore* database, even the *abstract* is private. To then classify the publication rightfully, the abstract is at least needed.

The user may want to import a publication from *IEEE* database, or may not. That's why the user needs to be able to import the publication using a file.

Import using a file

The idea is to build a *Javascript* module that will:

1. parse the given file, in *json* format, because the former dev on the site chose *json* as main file type, and because *ElasticSearch*, so *Crossref*, only gives *json* files.

The different file formats could all be parsed in *json* using external modules such as for instance:

- **Bibtex JS** to parse *Bibtex* files.

- **XML JS** to parse *XML* files.
- etc..

2. put the data in this *json* file, into the *MySQL* database, using, as the former dev used before me, **Sequelize**.

I assume that all this part of parsing should be done on the *client side*.

Import using a search engine

The idea is to use the *Crossref API*, to retrieve *json* files and then put it in the *MySQL* database using *Sequelize*.

I assume that all this part of parsing should be done on the *client side*, to retrieve the publication.

However, when the publication is retrieved, one important thing may be to also find the authors. So here the *Semantic Scholar API* could be used to retrieve data about the authors because this *API* has access to more than *79 Million* authors.

I assume that all this searching part should be done on the *server side*.

Besides, it could be possible to retrieve very cool analytics from *Semantic Scholar API*, such as **themes**, and even **AI-generated topics and sub-themes**.

EOF